NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

INTERCONNECTIVITY VIA A CONSOLIDATED TYPE HIERARCHY AND XML

by

Brian J. Lyttle
Todd P. Ehrhardt

December 2000

Co-Advisors:	Valdis Berzins
	Ge Jun
Second Reader:	Paul E. Young

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 2000	Master's Thesis

4. TITLE AND SUBTITLE
Interconnectivity via a Consolidated Type Hierarchy And XML

5. FUNDING NUMBERS

6. AUTHOR(S)
Lyttle, Brian J. and Ehrhardt, Todd P.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

ABSTRACT (maximum 200 words)

We propose building a software system that passes any message type between legacy Command, Control, Communications, Computer, Intelligence, Surveillance and Reconnaissance (C4ISR) systems. The software system presents significant cost savings to the Department of Defense (DoD) because it allows us continued use of already purchased systems without changing the system itself.

In the midst of the information age, the DoD cannot get information to the warfigher. We still maintain and use heterogeneous legacy systems, which send limited information via a set of common messages developed for a specific domain or branch of DoD. Our ability to communicate with one message format does not meet our needs today, though these stovepipe C4ISR systems still provide vital information. By combining these systems, we will have a synergistic effect on our information operations because of the shared information.

Our translator will resolve data representational differences between the legacy systems using a model entitled the Common Type Hierarchy (CTH). The CTH stores the relationships between different data representations and captures what is needed to perform translations between the different representations. We will use the platform neutral eXtensible Mark-up Language (XML) as an enabling technology for the CTH model.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Interoperability, Interconnectivity, Legacy Systems, XML, Consolidated Type Hierarchy, Information Systems	104
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFI- CATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

INTERCONNECTIVITY VIA A CONSOLIDATED TYPE HIERARCHY AND XML

Brian J. Lyttle
Captain, United States Army
B.S., United States Military Academy, 1992

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

Todd P. Ehrhardt
Lieutenant, United States Navy
B.S. San Jose State University, 1993

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2000

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION

In today's combat environment, the United States military and its allies find themselves in the midst of the information age they helped start. Information and systems that use information abound in all parts of the services and all locations on the globe. No longer can the side with the best trained and best equipped force be confident of victory. If an opponent can conduct efficient information operations, they have a significant edge. An important fact is that information operations take place throughout the spectrum of combat, from peacetime operations such as refugee relief to armed conflicts similar to Operation Desert Storm. This fact implies we will always conduct information operations, regardless of the place or time.

Information operations are "Actions taken to affect adversary information and information systems while defending one's own information and information systems." [DTIC] Information systems are normally the computer systems that receive, manipulate, and disseminate information. From this definition of information operations we realize these operations are both offensive and defensive in nature. An astute information operator could use propaganda in an offensive manner to destroy the public support of his enemy. Or, the operator could publish incorrect information about an operation in order to deceive

the enemy. Properly conducted, information operations are a powerful combat force multiplier that can significantly increase our ability to shape the environment and influence decisions at all levels of combat.

To influence decisions, commanders and their staffs need the most up-to-date information available. This information comes from many different sources, but especially from computer systems. The Department of Defense (DoD) developed many of these computer systems over the last few decades before interoperability became a concern. Often systems cannot pass information to each other because they use incompatible message sets.

One agency within DoD that tries to solve joint war-fighting problems is the U.S. Joint Forces Command (JFCOM). A subordinate element of JFCOM is the Joint Battle Center (JBC) in Suffolk, Virginia, which tries to resolve Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) issues, especially between the various information systems. Part of their C4ISR involvement is the assessment of new technology to solve interoperability problems between the services.

Many of the established information systems use message formats that possess a structured, though limited method of communication. Information is passed via a set of messages contained in a message set. These sets are rigid by design

and cannot be changed. However, one format cannot satisfy the needs of the entire DoD, not to mention our allies.

Commanders need all possible information in order to make accurate and timely decisions. The various information systems contain valuable data, but it cannot reach the commander because of incompatible data formats between information systems. Thus, there exists a need to increase the flow of information to the commanders, yet save development time and costs due to budget constraints. We believe DoD can continue to use the legacy systems if some method is developed that allows message passing between the computer systems.

We seek to design a format that bridges the differences between all the message formats called the Consolidated Type Hierarchy (CTH). The CTH is formed from all the message formats contained in the network of information systems, thus allowing a free-moving flow of information to all systems that desire it.

One new technology that has emerged recently is the eXtensible Mark-up Language (XML). With roots in the publishing industry (the Standard Generalized Markup Language), XML is now used by the e-commerce industry to allow interoperability between a variety of databases in a near-real time manner. Though these applications are business oriented, the application of XML shows great

promise in solving some of the DoD interoperability problems. We used XML to implement the CTH in our thesis.

By using the CTH model, we believe DoD can start integrating the legacy computer systems with significant cost savings. Our results on a small set of messages show the concept has promise and hope for interoperability.

II. CURRENT STATE OF AFFAIRS

One of the main difficulties in information operations is the task of getting relevant information to the user in the correct format. Many of our current systems are heterogeneous systems that do not communicate outside of their own format. Thus, we need the ability to share data with computer systems that were developed for diverse user communities with very different data needs and requirements. We are currently limited to sending text messages common to the various computer systems, and some systems cannot even do that.

A. A MEGAPROGRAM

We can think of attempts to continually use legacy systems and their information as an example of megaprogramming[GW92]. Megaprogramming is a concept developed by the Defense Advanced Research Projects Agency (DARPA) as part of an effort to reuse systems that already exist. A megaprogram is a software program that utilizes commercial off the shelf (COTS), and government off the shelf (GOTS) software systems as if they were modules. The modules, or megamodules as the authors call them, are internally homogeneous, independently maintained software systems managed by a community with its own terminology, goals, knowledge and programming traditions. We call the

concepts, terminology, and interpretation associated with each domain specific megamodule an ontology.

Unlike the distributed federated databases used in [GW92], our legacy system megamodules possess only the ability to export information through a set of standardized messages. This constitutes a key difference between tying together legacy systems and the megaprogramming previously envisioned. Megaprogramming relies heavily on databases to furnish the ability to import and extract data from the heterogeneous systems, whereas our system must rely on the information sources to push the information out. We have no mechanism to actively query or pull information from the source. This limits our ability to access information within the megamodule.

Because some systems cannot automatically extract data from a distant machine, they are reliant on other machines to send regular updates consisting of any new data they find. This feature is unfortunate because the remote systems are not always configured to meet the needs of the other systems. In some cases operator action is required to send and receive information from the source. System operators must then rely on standard operating procedures (SOPs) for regular updates of information outside of our local system. This does not agree with the mega-programming concept as stated in the paragraph above. This makes reuse

of legacy systems a limited example of mega-programming, but still useful.

B. MESSAGE FORMATS

In previous years, information systems defined a set of messages for each system. This set of messages contained the information most commonly needed by consumer systems, and was often domain specific. One common message format used by many systems is the United States Message Text Format (USMTF). The U.S. and our NATO allies used USMTF to increase our ability to communicate tactical and other information. The format of USMTF is well established, but its fixed field format wastes bandwidth by sending empty information. Because USMTF messages require larger bandwidth capabilities than most land forces possess, the land forces use variants of USMTF. USMTF may also provide more information than the destination system needs.

Coalition Information eXchange (CIX) is a newer data message format constructed by Defense Information Systems Agency (DISA) with more capabilities than the Over the Horizon Gold (OTH-G) message format used by the Navy and Marine Corps. However, unless the receiving system can translate from CIX, the information is unused and useless.

To communicate between different message formats such as CIX and USMTF, current implementations use software programs called translators. The translator alters a system

message from one legacy computer system format into another format for a different legacy system. The translator is implemented via a third generation language such as C++ or Ada. Providing some way for different existing systems to share data presents an opportunity to save significant development costs in the design of replacement systems built to share data. Enabling systems to share data also saves end-user time, since data does not have to be entered by hand from one format or system into another.

However, making translators is a time consuming task when constructed manually.[Sin98] The programmer must map the systems' message types, find corresponding messages, find data within the message that can translate between systems, and finally code the translator from scratch. Once completed, the translator only works from one message format to another specific message format. Although these translators are better than the manual transportation of data between systems, their creation is time consuming and of limited use. Each translator is expensive because of the specialized knowledge contained in the two systems. This also causes maintenance problems when the programmer leaves or a heterogeneous system changes its message format.

At this time, we do not possess an automated way of resolving representational differences between systems. Thus, the programmer must still complete the mapping by hand. We seek to construct a translator that uses a pre-

runtime developed framework to perform run-time message translation. This method would enable reuse of common translation routines, and would be able to translate messages among many different formats.

C. BACKGROUND RESEARCH

Part of our research revealed the similarities between integration of heterogeneous databases and legacy system integration. Since message formats share data among systems, we can consider messages to be results from a database query. Many current commercial databases share data between heterogeneous systems connected via networks. Reconciling differences between databases must be done over several levels.

At the highest level, databases must be reconciled over different schemas. Database schemas define the structure of the data, and how each piece of data is related to each other, how it's organized. The differences include resolving the representations between the tables found in each database.[HMS] This representational heterogeneity is defined as "variations in the meaning in which data is specified (for the data) and (the way it is) structured in different components". It is a natural consequence caused by creating independent data structures.[HM99]

The next level of reconciliation involves the naming conventions used in each database. A major cause of

conflict is the use of homonyms and synonyms. Homonyms use the same word for different concepts, such as "fire." In one context, the phrase results in artillery rounds impacting on a target, while in another context, the phrase summons the fire department. Synonyms describe the same object, but use different terms. Soldiers commonly use position and location to mean the same place.

Representational differences make up a third level for reconciliation. As shown in Figure 2-1, one community may define a geospatial position using the Military Grid Reference System, while another defines the position using a latitude/longitude representation. Both methods define the same real world object, but implement different methods and possess different attributes.

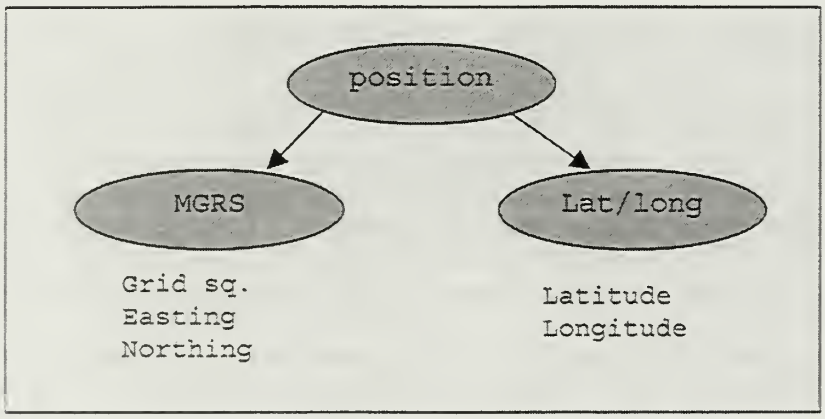

Figure 2-1 Different representations of the same location

Some other causes of differences in data representation include the low-level format of the data, such as precision or units of measurement.[KM98] Another cause is the range of values for a data type, which may vary from system to

system depending on the needs of the user and the hardware and software the user possesses. Older systems cannot represent larger numbers due to the size of the allocated memory or the processor used in the hardware.

D. PREVIOUS ATTEMPTS

Because of the many different systems and formats we are looking for a systematic way to construct translators, which opens the door to automation. This will save time, money, and results in more reliable communications.

In our search for a solution to the problem, we found several systems that try to achieve similar results.

One thing that almost all these systems or models have in common is the use of some kind of universal representation of data, or some universally agreed upon vocabulary. Most systems have these universally accepted terms and build on that in different ways.

1. Canonical Data Model

Roantree, Keane, and Murphy call their universal model a Canonical Data Model (CDM). This is similar to a universally agreed upon representation for a location. They introduced a model containing three layers. From top to bottom, the layers are: the Federation Layer, the Component Layer, and the Integration Layer. They use the lowest layer to isolate the effects of changes in a member database. The Integration Layer changes with the database in order to

maintain a consistent interface with the upper layers. Any time a change is made in the design or schema of a particular constituent database, its corresponding integration layer changes.[RKM]

2. Metadata

Another approach presented by Narinder Singh is to use metadata, which is information about data, to dynamically determine how to respond to a query. In this system, information providers must supply a description of the information they have to offer in terms of a standard vocabulary. This standard vocabulary is a list of universally agreed upon set of words, each word having a single meaning. Middleware provides access to the data sources. When a query is submitted from a user, the Tesserae Integration Engine dynamically creates a search plan and retrieves the information.[Sin98]

One drawback to this system is the time cost of creating a search plan on the fly. In a dynamic environment such as the web, the benefits would outweigh the costs; but, in our context there is no advantage to creating a search plan.

These previous methods have their merits, and we have tried to incorporate some of their achievements into our system. For example, it is apparent that in order to reconcile information from different databases, there has to

be at least some a priori agreement on what some of the terms mean. However, our context is different from the typical scenario in which databases are being integrated, since we don't have the ability to query data sources, and we don't want to assume the existence of a central data store.

E. RESPECTFUL TYPE CONVERSION

One of the most pertinent articles to our research is a paper written by Jeannette Wing and John Ockerbloom [JMJO00]. Their paper discussed the conversion of different types in such a manner that no data was lost. This pertains directly to interoperability because of the problems associated with data differences.

In their paper, Wing and Ockerbloom assume a normal subtype and supertype inheritance relationship, and call an instance of a type an object. The types follow what is known as the Liskov substitution principle, which is outlined in the article. The Liskov substitution principle says that the subtype inherits the attributes of the supertype, and an instantiated object of the subtype acts the same as the supertype when the supertype's method is invoked. A *respectful type converter* will convert two subtypes with a common supertype ancestor while preserving the behavior observable through the interface of the common ancestor supertype.[JMJO00]

Wing and Ockerbloom recognize type hierarchies may solve many interoperability issues by reducing the number of translators required from N^2 to 2*N translators.[JMJO00] They base their examples on an assumption that only one type will exist per file, which is unlikely to occur in our messaging system. A message may contain a position and a text message that have different supertypes. Unlike the paper, we must construct translators that contain many different functions because our messages will contain many different types.

Additionally, our system cannot actively retrieve information because of how the message systems are constructed. Rather, the information providers will push their data, as opposed to the data being pulled from its source. Therefore, a system that derives a search plan would not be appropriate.

F. THE EXTENSIBLE MARK-UP LANGUAGE (XML)

In order to construct our program, we needed a method that allowed us to express information in a manner independent of any platform yet still capture the meaning of the data. We found the eXtensible Mark-up Language (XML) met these criteria. Since XML is a fairly new language, we searched for current examples that utilized XML commercially and in DoD. In order to understand these examples and our thesis, we must first explain what XML is.

1. Meta-Language

XML is a meta language, which means it describes the data contained inside an XML document. XML separates the content of the document from the presentation of the data, which enables more programs to read the document.[PROXML] The separation occurs because XML only provides the means to describe the data, leaving presentation of the data to the receiver.

Mark-up tags surround the data in an XML document. The tags are very similar to Hyper-Text Mark-up Language (HTML) tags, with an important exception. While XML tags may use all but a small set of characters, HTML tags are predefined and restrictive. Unlike XML, the HTML language possesses functions that tell an HTML browser how to display the data.

Figure 2-3 is an example of how an XML document could describe a person. Note the document root mark-up tag entitled people, and how it surrounds the nested elements.

```
<people><!--This is a comment block-->
      <person>
         <firstName>Brian</firstName>
         <middleName>John</middleName>
         <lastName>Lyttle</lastName>
      </person>
      <person>
</person><!--This is an empty person element using open and close
           tags-->
</people>
```

Figure 2-3 Sample XML Document

2. XML Trees

XML works by forming a tree from the data contained in the XML document. The document must possess a root node in order for the parser to construct a tree from the elements within the document. Elements may be nested repeatedly beneath the root node, and may contain duplicate element names at the same level within the tree.

XML contains a powerful concept called a namespace that effectively allows homonyms. The namespace allows the writer to use the same name but with different associations, provided the writer distinguishes the namespaces. This allows the transformations and formatting functions at each viewer's platform to take the appropriate actions when parsing the document tree.[PROXML]

3. Parsers

In order to take actions on an XML document, we must be able to construct the tree in memory. The software program that constructs the document tree is called a parser. It is not responsible for presenting data to the user, unlike HTML. The parser ensures the document is "well-formed", which means the document obeys the XML syntax rules. XML parsers are powerful tools freely available from several sources. Both Internet Explorer 5.0 and Netscape's Mozilla 6.0 contain XML parsers in addition to HTML parsers. The IBM Apache Group (http://www.apache.org) wrote and provided

the source code for their Xerces processor for anyone to utilize for free. The Xerces parser is written in both C++ and Java, and is available for a variety of operating systems to include Windows, Linux, Unix, AIX, and Sun Solaris. The Xerces parser is the official parser of the World Wide Web Consortium (W3C) at this time, and is fully compliant with the approved W3C recommendations. It does not expand upon the approved requirements of the W3C for XML.

4. Validation

All of the parsers mentioned above are examples of a validating parser. Validating parsers verify the XML document obeys more stringent rules than the generic XML syntax. These rules are specified in a Document Type Definition (DTD) or a Schema. DTDs and schemas allow us to specify rules about what elements may appear in a document, the structure of the tree, and to a limited extent, what format (e.g. the order and number of occurrences) the elements must follow. DTDs and schemas serve the same purpose. They were designed to facilitate content checking, to some degree. Obeying the DTD ensures all users of our namespace can read our document using the same standard. The DTD is a W3C recommendation; schemas are only a W3C candidate recommendation. According to the W3C, "a Candidate Recommendation is work that has received

significant review from its immediate technical community. It is an explicit call to those outside of the related Working Groups or the W3C itself for implementation and technical feedback." Also, "a Recommendation is work that represents consensus within W3C and has the Director's stamp of approval. W3C considers that the ideas or technology specified by a Recommendation are appropriate for widespread deployment and promote W3C's mission." [W3C] However, schemas were designed to make up for some of the shortcomings of DTDs; and tools that support schemas are already on the market.

Schemas have several advantages over DTDs. Schemas allow open content models. An open content model provides extensibility to a schema. This means that I can reuse someone else's schema. If their schema doesn't contain all the elements I want to include in my schema, I can add elements. This allows greater reuse of schemas. Open content models are optional; however, and a closed content model can be specified in a schema if desired.

Schemas also provide some support for data types. Data types can be specified for elements and/or attributes. Beyond the typical data types found in common programming languages, the following data types are some of those supported: string, id, idref, nmtoken, nmtokens, entity, entities, enumeration, and notation.

Other advantages of schemas [MSDN]:

- Greater specificity of the number of occurrences of an element.
- Ability to specify if sub-elements must appear in a certain order.
- Accessible from Microsoft's Document Object Model.
- Schemas are well-formed XML documents (unlike DTDs, which have their own syntax).

We believe that although schemas are relatively new, their additional capabilities provide them a substantial advantage over DTDs. We recommend the use of schemas.

5. Transformation

If two users have different formats for their data, like many Defense organizations, we can transform the XML document using the eXtensible Stylesheet Language Transformation (XSLT). XSLT enables us to translate between vocabularies as well as merge existing resources. We can determine the correct stylesheet to use at runtime to dynamically translate between documents. We do not have to write procedural language code for most applications, although it may be necessary in some cases.

Stylesheets provide a major contribution toward achieving our goals. They are a part of the XML world, and as such, share many of the same benefits. They can be transferred using the ubiquitous hypertext transfer protocol (HTTP). They can be applied to XML documents by the XML

processors. The XML processors are COTS, and are available for free. Stylesheets can also refer to other stylesheets. Therefore, they can be used and reused in a modular way, also providing cost savings.

Internet Explorer 5.0 and the MSXML 3.0 parser allow the programmer to write procedural JavaScript functions in order to assist with transformation. We have not found any other free commercially available parsers that allow us to do this in a packaged format, though we can construct a parser from source code like Xerces and write functions in the same manner.

However, this requires a compiler for each target machine for the functions each programmer may write. Parsers perform much of the work contained by the XML language, and a good working parser should not be modified greatly. The commercial parsers such as Internet Explorer and Mozilla provide the functionality we need for this demonstration.

III. XML USAGE EXAMPLE SYSTEM

A. THE JBMI EXPERIMENT

One organization with XML experience is the Joint Battle Center (JBC) based in Suffolk, Virginia. JBC is part of Joint Forces Command (JFCOM), and is charged with finding joint solutions for Command, Control, Communications, Computer, Intelligence, Surveillance, and Reconnaissance Systems (C4ISR) inter-operability. In order to fulfill this mission, they conduct experiments with several organizations each year.

We witnessed Phase Two of an experiment entitled the Joint Battle Management Initiative (JBMI). JBMI sought to prove XML is a valid technology for improving inter-operability and inter-connectivity between systems. All four services provided computer systems for the experiment.

JBC defined two different levels of sharing information between systems in accordance with the Defense Information Infrastructure Common Operating Environment (DII COE). Interoperability at its highest level allows systems to import and export information as if the remote site were actually part of the user's system. Inter-connectivity is several steps lower, and allows systems to pass limited messages between different systems.

The computer systems at JBMI accurately reflected the problem in DoD today. The primary system was the Global Command and Control System (GCCS), which controls high level operational units across DoD. It specifically targets units the equivalent of an Army Brigade level or higher. It utilizes CIX as its means of message passing. The Navy and Marine Corps also sent their versions of GCCS, which are compatible with the other services' GCCS systems.

The U.S. Army provided a system entitled the Advanced Field Artillery Tactical Data System (AFATDS). AFATDS is a member of the Army Battle Control System set, and is the command and control system for all ground fire-support systems in both the Army and Marine Corps. AFATDS also interacts with our English and German allies using its own specific format developed many years ago. It can send and receive a limited number of USMTF messages.

In an interesting twist, JBC integrated two devices currently available on the commercial market. The first was a Palm Pilot V, which is a personal digital assistant. JBC programmed the simple USMTF Call for Fire and Observation Report messages into the PDA. They programmed the same ability into a cellular telephone, and communicated using the Wireless Application Protocol to the networked systems.

All the systems connected via a hardwire LAN into a web server. The web server allowed each unique system to subscribe to a message set or an individual message type

from the USMTF. As each legacy system produced a message, a software wrapper transformed the message into an XML formatted message. It then sent the XML mark-up message to the web server.

The web server received the message and removed the XML mark-up from the message. It parsed the message to discover the USMTF message type. The server then found a data directory specific to that message type, and saved the message. A Visual Basic monitor script periodically checked the directories for new information. If the monitor found new information, it checked a database to discover subscribers of that message type.

If a subscriber was found, it called upon functions constructed in Java code to transform the message into the appropriate type. If the destination system required the message in the HTML format, the XSLT processor was called to make the conversion. Most systems subscribed for an HTML representation of the USMTF message or email.

This system allowed the cell phone user to send a Call for Fire message to the AFATDS system via the web server. The AFATDS equipped unit could then provide indirect fire support onto the target. It also allowed the GCCS system to update its database, and the Air Force TCDB to enter the target information for use in plotting aircraft routes or further intelligence usage.

Other abilities included at this demonstration were comma-delimited files used in spreadsheets and word-processing documents. Since many of our allies do not have the funds required to make military specific information systems, they must rely on Commercial Off The Shelf (COTS) products.

An extremely useful application of COTS and XML was the target list used in the joint targeting process. Using AFATDS, a message containing a target list was sent to the web server. Upon receiving the message, the JBMI engine found the coalition subscribers that wanted a copy of the list. The engine translated the target list into a spreadsheet file, and sent it to the destination machine via email. Though the system lacked security restraints, it demonstrated the ability of XML to send various messages using COTS equipment.

Given the accomplishments of the JBMI engine, we knew XML presented a means to accomplish interoperability between systems. It allowed messages to transform from native legacy format into XML and then be used in a different system. However, the engineers were required to write source line code in Java to accomplish this. We believe using XML and other COTS tools along with a different methodology can accomplish interoperability between systems cheaper and faster than writing source code.

B. ASSUMPTIONS

We made several assumptions in our thesis. We assumed all the messages we received were well-formed XML documents and complied with a DTD for that specific message type. We assumed this because each system should send messages in the correct format, else it would not be fielded to the force. The parser would not read messages with incorrect formats because it would fail the validity check when a stylesheet or a DTD was applied to it. In a fielded system, a failed message would be returned to the sender with the appropriate error message. This service would take a small amount of time, and not impact the performance of the system. Additionally, we did not think we needed to check for transmission errors because the TCP/IP protocol stack conducts those error checks for us.

In our environment, we assumed an experienced software engineer would use the system. The messages will depart and arrive in an XML mark-up format of the original system message.

While we knew the translator system could be implemented either in a point-to-point system or in a publish/subscribe architecture, we chose to implement the point-to-point system. Although not as robust as the publish/subscribe architecture, the point-to-point implementation is sufficient as a first step for a proof of

concept. The point-to-point implementation can then form the basis for subsequent implementations. In the point-to-point system, each system possesses a copy of the translator and a means of communicating to the other system.

We assumed individual systems using this software would possess similar capabilities to our own, because our demonstration is based on the systems used by JBC during the JBMI exercise. That is, it would be a machine using Windows 95, Windows NT, or Windows 2000.

Given these assumptions and requirements, we can now describe the design of our system.

IV. THE CONSOLIDATED TYPE HIERARCHY

As we introduce you to the Consolidated Type Hierarchy (CTH), remember our goal: we are trying to achieve interoperability between legacy systems that have different views and representations of data. Our general approach is to set up a common framework that we can use in matching data sources with potential consumers. Translations will be defined in terms of the framework before run-time, and will be applied at run-time. Since the legacy systems we have in mind traditionally have shared their data through messages, we will consider the message formats they use rather than the data stores internal to the systems themselves. Before we explain what the CTH is, we will discuss what we need in order to create a CTH, the environment.

A. THEORY

1. System Schemas

Schemas provide a blueprint for the data to be shared. They can be thought of as Application Programmer Interfaces (APIs). Each message format will have its own schema. It is our way of knowing what data is contained within and provided by that data source or consumed by that recipient.

If we only had to be concerned with converting between two message formats, we could easily map data fields from one message format to the other. This simplified problem

would be trivial and not warrant further effort. However, as more formats are considered, the task becomes more complicated and requires considerably more work. If you had N different formats to reconcile with each other, N^2 direct mappings would be required.[JWJO00]

2. The Global Schema

The global schema is a global view of the data to be shared. It provides the context for data to be shared among systems. The elements of the system schema have a "kind-of" relationship with the elements of the global schema. For example, one element in the global schema might be a location. Although latitude-longitude and MGRS coordinates have different formats, they are both a kind-of location. They convey the same information.

The real purpose of the global schema is to capture the structure of composite types. If we were to send a list of locations, it would be meaningless. We must put information in its context. In other words, a position is an attribute of some other thing, like a ship, a tank, or an aircraft route. The global schema captures the contexts in which it is used.

3. Consolidated Types

Every element within the global schema is a consolidated type. In the example mentioned above, location

is a consolidated type and latitude-longitude and MGRS coordinates are legacy system subtypes.

Consolidated types are more than just an abstraction. Consolidated types must have a concrete representation in order to gain the advantages offered by having them. It's important to consider the physical representation of a consolidated type with care. Consolidated types are derived from pre-existing subtypes that are to be reconciled. Therefore, one method of choosing a representation would be to adopt the representation of one its subtypes. However, we would like to be able to convert from a subtype to the consolidated type and back to the same subtype without losing any information. Consequently, is important to select the representation with the highest degree of precision.

4. The CTH

The global schema represents a global view of information that is to be exchanged. It is a bridge format, which reduces the number of translations that must be defined. The elements of the global schema are consolidated types. The CTH does more than describe the structure of the global schema. It also contains the relationships between its elements and the elements of its constituent schemas. We introduce a separate term for the consolidated type hierarchy because neither the global schema nor its

elements capture both the structure of the consolidate types and their relationships with the elements of the various system schemas.

Now that we have explained the theory of the different parts and their relationships, it's time to look at how we implemented and integrated these pieces.

B. IMPLEMENTATION & EXAMPLE

We have created a simple example to illustrate how the different parts of our system fit together to achieve the desired result. In our example we have two message formats that we want to reconcile. We invented the message formats for the purpose of this example, but they are adequate to show the relationships between the different parts of our system and how they are used.

Both formats carry information about tactical units in a battlespace. The Army message format is designed to contain information about ground forces. Originally constructed as a voice message, it is now a standard digital message as well. The Navy message format contains data about ships sent via tactical data links from a variety of sensors. Both messages contain information about objects the operators are observing.

1. Schemas

The schemas were simply implemented as XML schemas. For our purposes, the essential requirement was to be able

to capture the structure of the data. This could have been done in many different ways, including UML diagrams. However, since DTDs or XML schemas can also be used for validating the XML documents, they might already exist for some systems and they could serve a dual purpose. We prefer the use of schemas over DTDs for reasons given in chapter 3, and our example uses XML Schemas.

Before we go further, we'd like to acknowledge a valuable tool we discovered in our research called XML Spy. XML Spy is the product of Altova GmbH, of Austria. It is an easy to use integrated development environment for XML, with authoring tools for XML documents, DTDs, schema, and style sheets. The product is available for download at www.xmlspy.com and free thirty day trial downloads are available. We used XML Spy for all the XML and related coding for our examples. We have included partial screen shots of the program in Figures 4-1 through 4-3 below. We are using the program to show the schema, because it can display them in a graphical representation, rather than having to look at the code; however the code is included in the Appendices.

The Army message format we called a SALUTE message. Figure 4-1 depicts the schema for the message format. The root element in the SALUTE schema is the element named SOURCE. The Type element contains information about the message type, and the GroundUnit element contains the

information on the ground units. Note the symbology depicts that there can be a sequence of GroundUnit elements contained in a valid XML document.

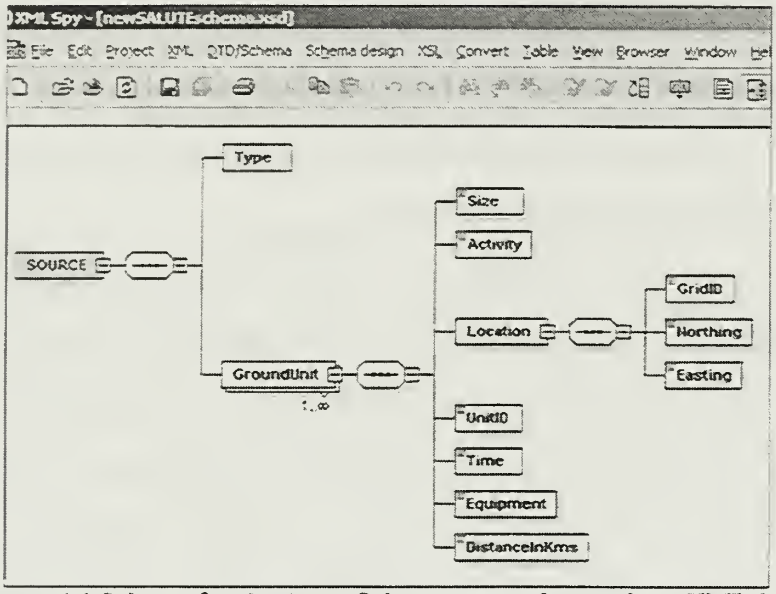

Figure 4-1 Schema for the Army Salute message format from XML Spy

Figure 4-2 depicts the Navy message format. It has some fields that will map to the Army message format, and some that do not.

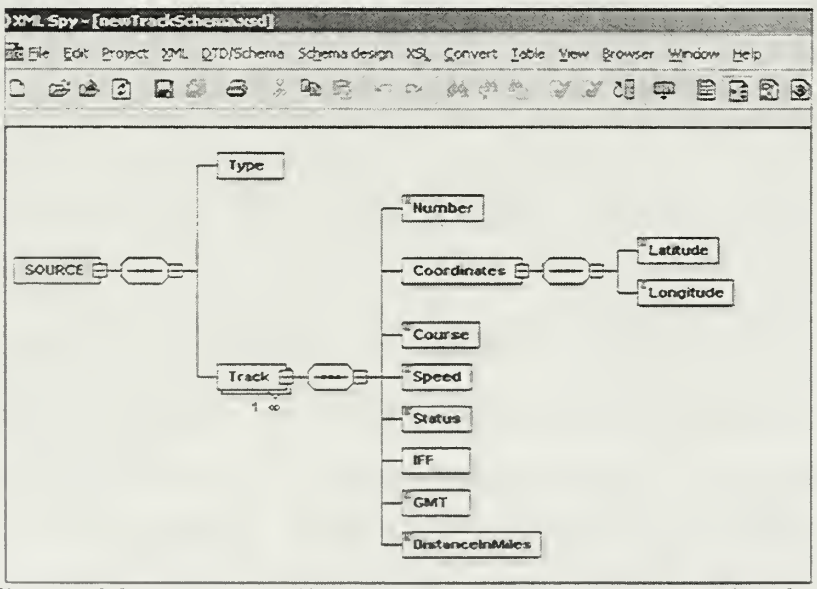

Figure 4-2 Schema for the Track Report message format from XML Spy

The global schema in Figure 4-3 depicts a composite view of the information provided by both message formats. Here you can see that Location is a consolidated type.

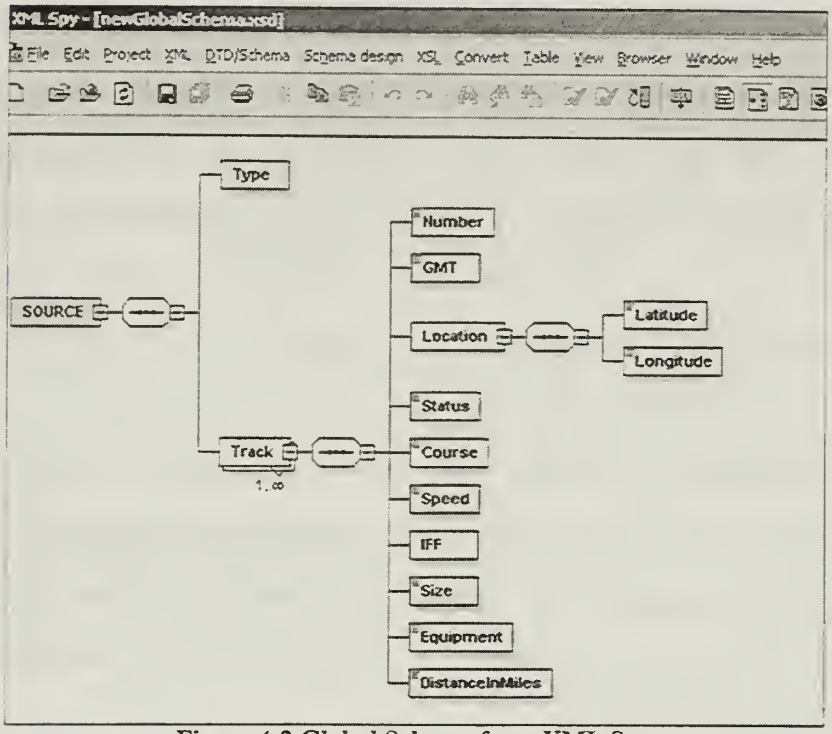

Figure 4-3 Global Schema from XML Spy

Also, notice that we included elements in the global schema, such as Course and Speed, which did not have a corresponding element in the Army schema. If a Navy system were to send a message to an Army system, the Army system has no use for such information. This begs the question, why include these elements in the global schema?

There are two reasons to include those elements in the global schema. The first reason relates to the comment we made earlier about choosing the representation with the greatest precision. If we convert a Navy message to conform

to the global schema without those elements, we would lose the Course and Speed information in the process. If we then convert it back to the Navy message format, we can't get that information back. We threw away that information. We would like to be able to convert from any system format to the global format and back without losing any information.

The second reason to include unique elements in the global schema is to make it easier to find compatible elements between schemas. Imagine that we decide to integrate a third message format into the global schema, and we left out Course, Speed and other elements unique to each of the preexisting Army and Navy schemas. If the new schema we want to introduce has elements that do correspond to the previously unique elements, we may never discover the correspondence, unless we also look for corresponding elements in the Army and the Navy message schemas. Instead, if we include all of the elements, then when we integrate a new schema, we will be able to discover the common information to be shared among systems, without having to analyze each system independently.

2. Consolidated Types

We captured the consolidated types in an XML document we named CT.XML. Pictorially, you can think of CT.XML as shown in Figure 4-4. Each root node represents a

consolidated type. Each child node depicts the corresponding element from a particular message format.

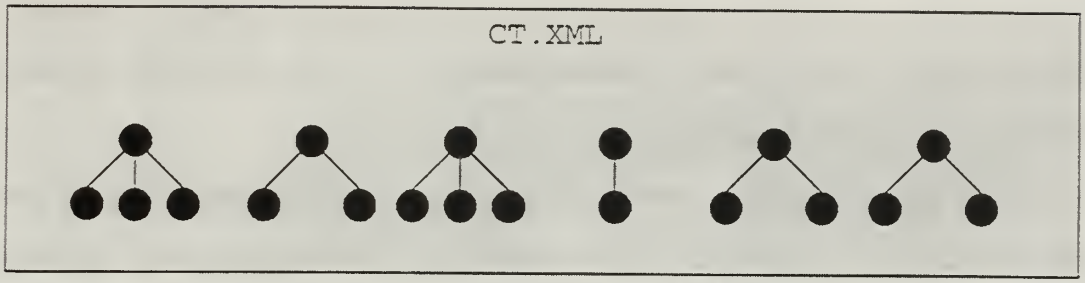

Figure 4-4 Symbolic view of CT.XML

Figure 4-5 is an excerpt from CT.XML, the XML representation of the consolidated type hierarchy. The full listing is included in Appendix F.

```
<Location>
   <TrackReport name="Coordinates"/>
   <Salute name="Location"
     upXlate="Grid2LatLong.xsl"
     dnXlate="LatLong2Grid.xsl"/>
</Location>
```

Figure 4-5 The consolidated type Location from CT.XML

Figure 4-5 shows how the consolidated type, Location, is entered. The outer-most element is the name of the consolidated type, which comes from the global schema. The nested elements name the message formats that have a kind-of Location. Since both track report messages and salute messages have attributes that are a kind-of location, they are both listed here. Each of the nested elements may have between one and three attributes. The name attribute specifies the name of the corresponding element in their respective message formats. The upXlate attribute contains the name of the style sheet that will translate from the

enclosing message format to the format of the consolidated type. The style sheet named in the dnXlate attribute will perform the reverse operation, taking an instance of a consolidated type, and transforming it to conform with a specific message format.

Like many other aspects of our implementation, there were alternate ways of implementing the mappings between message formats and the global schema. One disadvantage of the way we implemented it is that searching through CT.XML for the translations would be slow compared to other methods, such as a table lookup or database query. But, since CT.XML will be searched when the stylesheets are generated, which happens prior to run-time, the speed of the search will not affect run-time performance.

C. CTH USE

Figure 4-6 shows a conceptual view of the CTH. The Army schema is in the upper plane, and the global schema is in the lower plane. The dashed arrows represent the associations and the translations between elements in the global schema and the Army schema, information that is stored in CT.XML. We have only included the Army Salute schema in the figure in the interest of readability, but we could have presented another plane for the Navy Track Report schema as well.

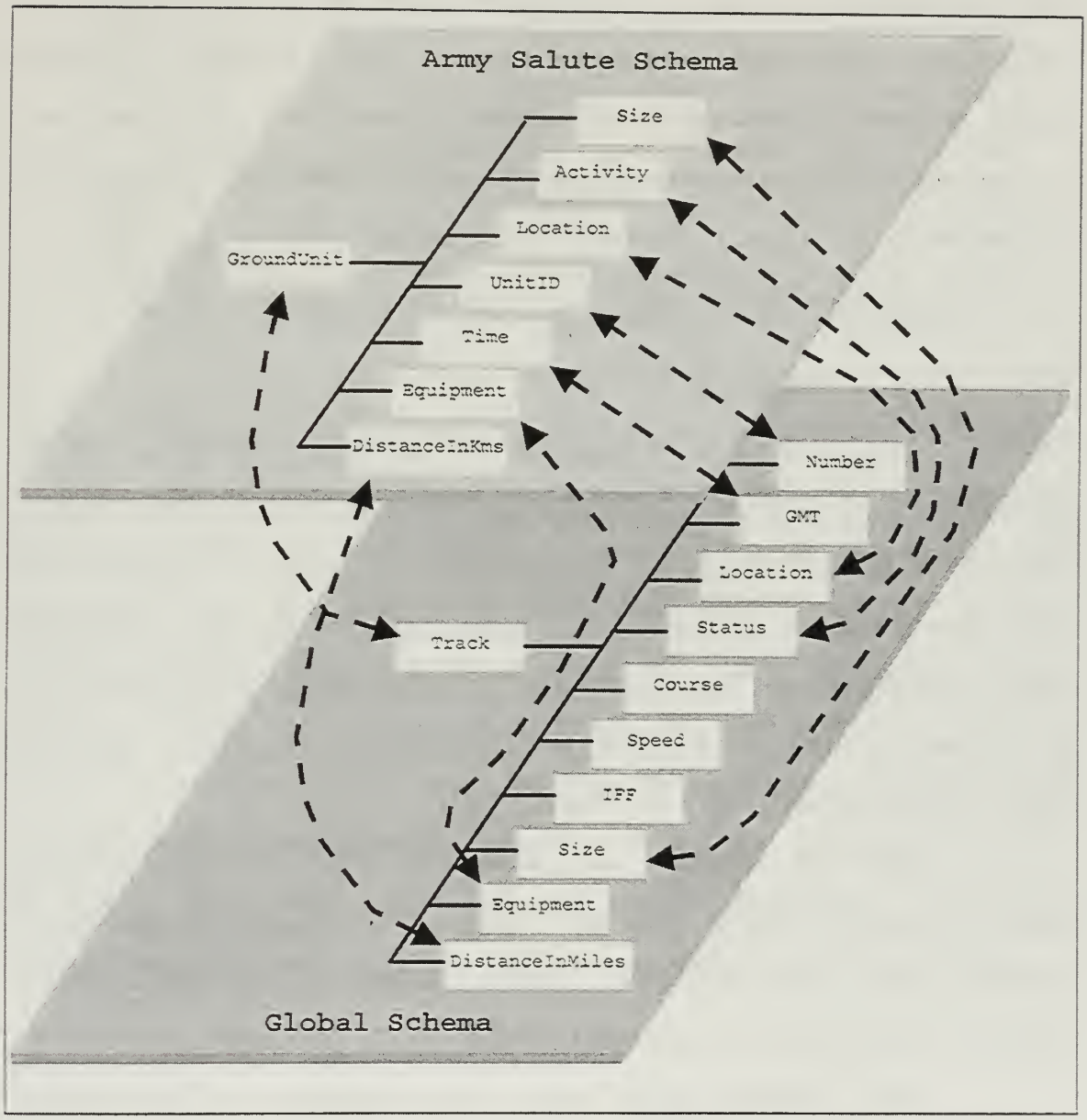

Figure 4-6 Conceptual View of the CTH. The Army schema is in the upper plane, and the global schema in the lower plane.

This is all we need to have a translator. When a translator receives a message it could determine the format, then recursively apply translations defined in the CTH by the arrows. Currently we create stylesheets before run-time based on the information contained in the CTH. At run-time we let the XSL processor act as our translator using the stylesheets to give it processing instructions.

1. Before Run-Time

a) Mapping

The CTH is a framework for matching potential data sources and consumers. It enables the sharing of that data, despite representational differences. When a system is introduced into a network, a schema for the data it exports and/or imports must be available or must be produced so that its elements can be mapped to the global schema. In our work, we performed this by hand.

In our system we generated the initial global schema from the Navy schema. Then we integrated the Army schema into this initial global schema. We will walk through the steps we followed during this process.

We started with the root element in the Army schema and looked for a corresponding element in the global schema. We descended through the structure of the Army schema, establishing these correlations at every level

possible. When we mapped the Army Schema to the global schema, we established these relationships:

Army Schema Element Name	Global Schema Element Name
GroundUnit	Track
Size	
Activity	Status
Location	Location
GridID	
Northing	
Easting	
UnitID	Number
Time	GMT
Equipment	
DistanceInKms	DistanceInMiles

Table 4-1 Initial Mapping of Elements in the Army Schema to the Global Schema

As you can see, Size and Equipment in the Army schema did not have corresponding elements in the global schema, so we added them to the global schema and we add them to CT.XML as consolidated types. GridID, Northing, and Easting also did not have corresponding elements in the global schema; however, we did not add those elements to the global schema as we did with Size and Equipment. This is where an engineer will have to decide whether to incorporate the elements into global schema, or define a translation at a higher level that will perform the conversion. Table 4-1 shows the mappings between the two schemas at this stage.

Army Schema Element Name	Global Schema Element Name
GroundUnit	Track
Size	Size
Activity	Status
Location	Location Translations: Grid2LatLong.xsl LatLong2Grid.xsl
GridID	
Northing	
Easting	
UnitID	Number
Time	GMT
Equipment	Equipment
DistanceInKms	DistanceInMiles

Table 4-2 Initial Mapping of Elements in the Army Schema to the Global Schema

b) Translating

When the mapping is complete, the engineer needs to determine which of two types of translations are required. The two types of translations are those that consist of nothing more than an element name change; and those that require a change in the data. Since XSLT facilitates modularity, some of the latter types of translations might already be defined. In our example, we defined translations that converted from grid to lat-long and back, and made the appropriate entries in CT.XML. Figure 4-5 shows the CT.XML entry for Location. Although our stylesheets do not actually convert a grid position to a latitude and longitude position, the intent here is to outline the process of reconciling a schema with the global schema.

Once each element's translation is defined, a pair of stylesheets can be generated that will translate from the particular message format to the global format, and back down, as in Figure 4-7.

Let's look at one of the stylesheets to see how the translations are defined in XSLT and how the process of

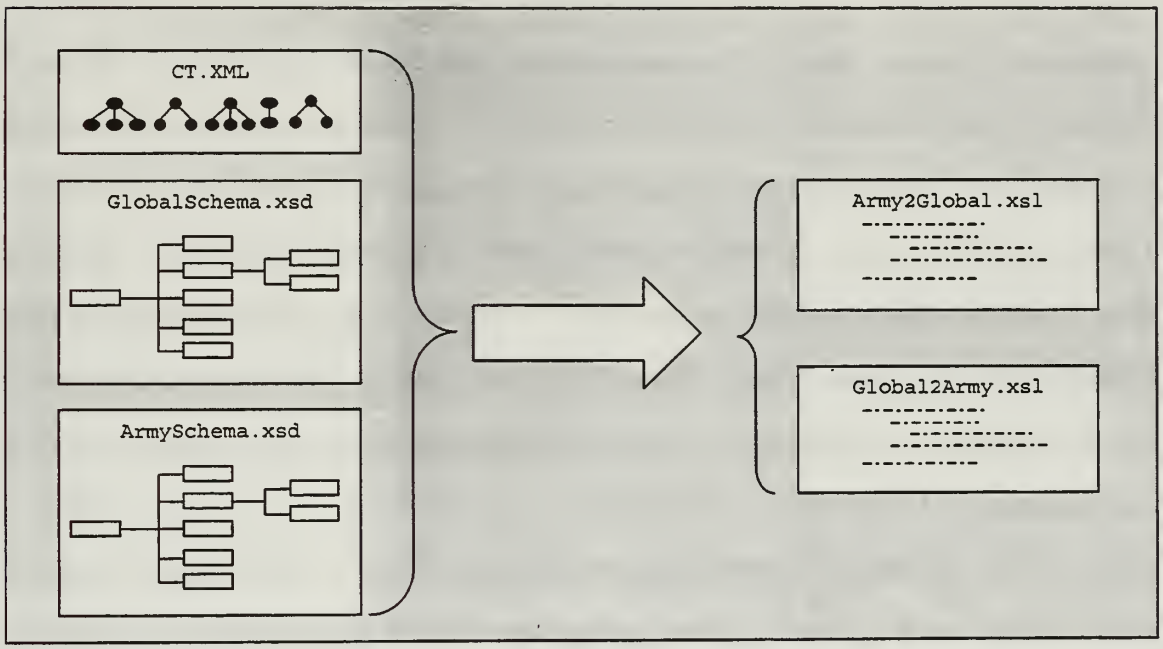

Figure 4-7 Generation of the Stylesheets

generating the stylesheet could be automated once the mapping has been completed. *(This explanation assumes the reader is somewhat familiar with the way that stylesheets work.)*

Our example comes from Appendix G, which is a stylesheet that transforms an Army SALUTE message (Appendix A) into the global CTH format. The first significant

instruction is on line 9. Line 9 tells the processor to look for an element named SOURCE in the XML document to be translated. We used SOURCE as a root node that would be common to all schemas, or message types. Nested in the SOURCE element is the element named Type, which we also used as an element common to all message formats. They serve as an identifier for the source and message type. Lines 10 through 13 are what the processor will output when a SOURCE element is found by the processor. Line 11 is significant because it specifies the schema that the output XML document must conform to, GlobalSchema.xsd. Given that the SOURCE and Type elements are standard elements in all messages, and given the schema for the output message, an automated stylesheet generator could produce this code in a stylesheet.

Lines 18 through 30 tell the processor how to translate a GroundUnit element. They tell the processor that the equivalent name in the global schema is a track, and they specify the order in which to process the children of the GroundUnit element. It is important for the sub-elements to appear in the output document in the correct order so that the document conforms to the global schema. Notice that the order of the output elements is specified in terms of the source schema element names, except lines 24 through 26. Those lines correspond to elements in the

global schema that have no equivalent element in the Salute schema.

A program could automatically generate this XSL code as well. The name correspondences between the schemas' elements are contained in CT.XML. The order in which the sub-elements should be processed is specified in the output schema, in this case the global schema.

Recall that earlier we said there are two basic types of translations. One type of translation merely involves a name change, and the other translation involves a change in the data. Most of the translations contained in Army2Global.xsl are of the former type. However, the translation from MGRS coordinates to latitude/longitude coordinates does require a change in the data. Line 5 is an import instruction to the processor. When the processor sees line 5, it effectively reads the stylesheet Grid2LatLong.xsl and pastes it in place of the import statement. Again, the information required for this line is contained in CT.XML. Incidentally, we chose to use the import statement to demonstrate modularity of stylesheets; however, we could have just done the copy-paste operation ourselves, or a program that generates the Army2Global stylesheet could do it.

We used JavaScript to perform the conversion from miles to kilometers, but we were unable to use the import functionality of XSL because of it. We'll discuss those

efforts later in this chapter. For the present discussion our aim has been to show the content of Army2Global.xsl, and that it could be generated automatically.

2. During Run-Time

Sending a message from System A to System B involves two translations. The first translation will transform the message from System A's format to the global format, the upward translation. The second translation will convert from global to System B's format, the downward translation. Both translations could be performed on either side of the transmission, as long as they're done in the proper order. That is, both could be done by the sender's translator, both by the receiver's translator, or one on each side.

There are two basic problems with doing both the upward and downward translations at the source. First, the source translator would have to know who all the recipients are, along with the appropriate translation for each. It would perform the upward translation and then it would have to perform downward translations for every different type of recipient, and send out multiple versions of the same data. The second potential problem is that changes in a consumer's schema might require the use of a new stylesheet that performs the new downward translations. Now we have to

worry about how to disseminate the new stylesheet to every source that produces information for the modified consumer.

The problem with performing both upward and downward translations at the consumer is essentially the same as the second issue, above. We must have a method of disseminating changes in a producer's upward translations to each of its consumers. Furthermore, both methods would involve some kind of lookup table that would be used at run-time in order to identify the appropriate stylesheet to apply to an outgoing or incoming document.

It is much simpler however, to perform the upward translation at the source and the downward translation at the receiver. This implementation eliminates the complications posed by the other two. Only one version of the document has to be sent. No lookup tables are required because producers always apply the same upward translations to their outgoing messages, and consumers always apply the same downward translations to incoming messages. Also, changes to producer and consumer schemas are localized. Figure 4-8 is a collaboration diagram showing how the system would work.

The CTH will not solve every problem by itself. Translations will still have to be written for many conversions between consolidated types and data contained in specific message formats. What the CTH will do for us is vastly reduce the number of translations that must be

defined, and in some cases enables reuse of those translations. It may also provide a framework for semi-automated generation of the translations.

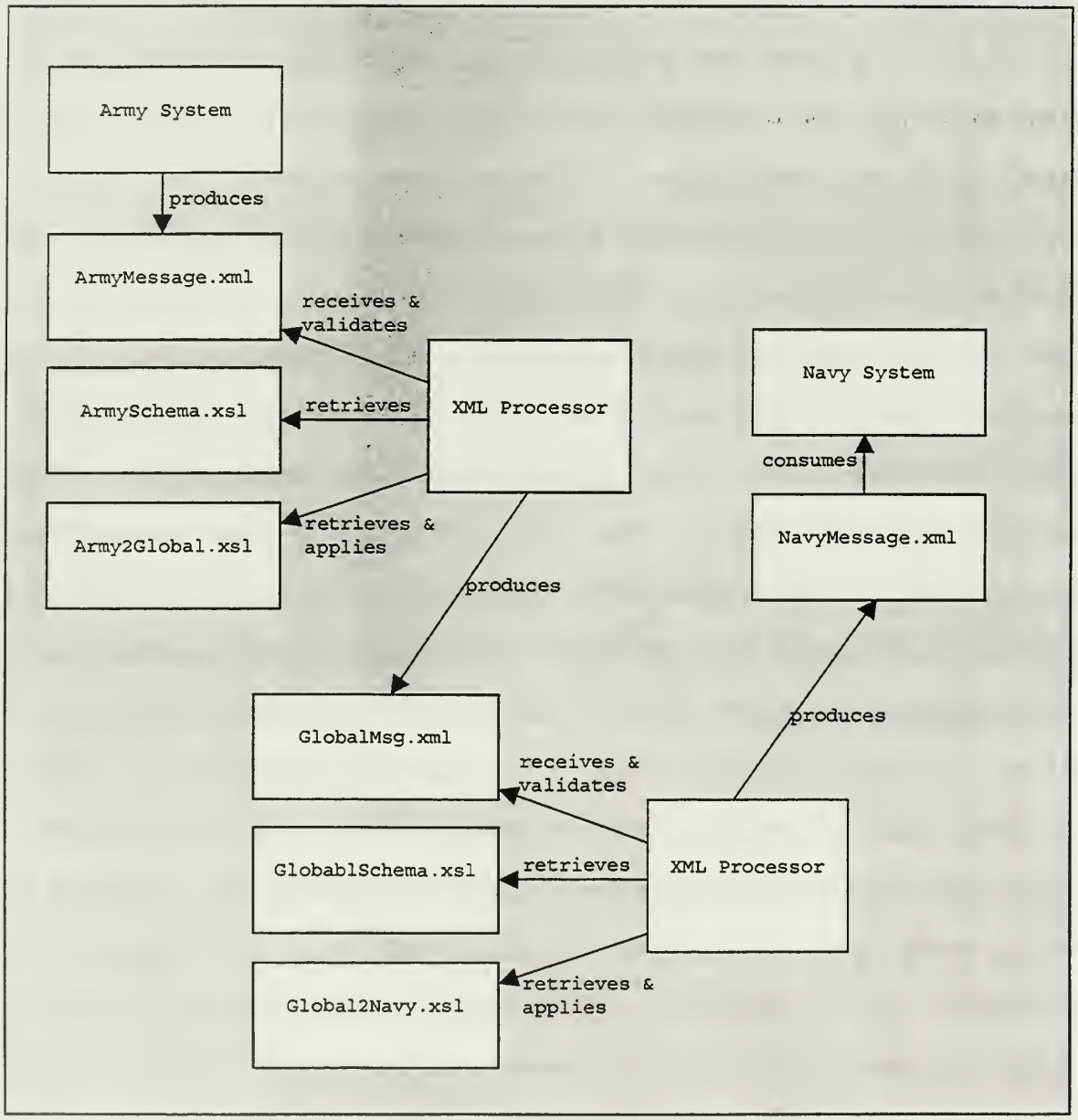

Figure 4-8 Collaboration Diagram of Proposed Implementation

D. RESULTS

We tested our system using a series of steps, incrementally checking what's been advertised about XML against what we were able to achieve. We started by creating two XML documents, one to represent a fictitious Army message format Appendix A, and the other, Navy, Appendix B. We created schemas for them, Appendices C and D, respectively. Next, we created a global schema that incorporated elements from both message formats, Figure 4-3 and Appendix E. Then we created CT.XML, Appendix F, to show the relationships between the elements of the global schema and its constituent schemas.

After entering correspondences between the message formats within CT.XML, we created the stylesheets to translate from the Army SALUTE message directly into the Navy Track Message. The main goal at this step was to verify the performance of an XSL processor. To execute the translations, we used a freeware program named Xalan constructed by IBM Apache Group (http://xml.apache.org/). Xalan is an XSL processor written in a variety of languages for different operating systems. The program takes command line parameters to specify the input and output XML documents, and which stylesheet to apply. The program and the stylesheet worked, and we also found that the resulting

message conformed to TrackSchema, which is the schema defined for the Navy Track Message.

Our next step was to create four stylesheets that performed the upward and downward translations for both Army and Navy message formats. We wanted to test the ability to translate from Army to Navy via the global schema, and perform the reverse. We also wanted to test the modularity of the stylesheets; so, we created two more just to handle the translation of positions, going from MGRS format to latitude-longitude format.

However, translating from MGRS to latitude-longitude requires the use of capabilities the W3C implementation does not support. Functional code is required in order to perform calculations on the data contained by an XML document. The Microsoft implementation of XSL supports JavaScript and Visual Basic Script (VBScript) functions that provide this capability. It uses the xsl:eval statement to invoke script functions from those two languages, but it does not support the import or include instructions as outlined in the W3C XSL namespace. [MSDN2] We implemented some of the final stylesheets (Appendices I and Q) using the xsl:eval processing instruction to demonstrate that XSL is capable of invoking a functional transformation for a user's specific needs, such as converting miles to kilometers.We converted the miles element into kilometers using JavaScript's math library. The stylesheet invokes the

commands using xsl:eval, which then searches for the language, specified in the second line of the stylesheet, as in Appendix I. Since this is an ability that Microsoft implemented for their own XSL processor, MSXSL [MS], the Xalan processor does not process the xsl:eval command. Table 4-3 is a listing of all the files we used, and their purpose.

Appendix	File Name	Description
A	ArmyMessage.xml	Message generated by an Army system. Valid in accordance with SALUTEschema.xsd
B	NavyMessage.xml	Message generated by a Navy system. Valid in accordance with TrackSchema.xsd
C	SALUTEschema.xsd	XML schema for validating messages generated by an Army system.
D	TrackSchema.xsd	XML schema for validating messages generated by a Navy system.
E	GlobalSchema.xsd	Contains the global view of data to be shared. Puts consolidate types in context. Also used for validating messages translated into the global schema.
F	CT.xml	Contains the relationships between the elements of the global schema and the elements of the Army & Navy schemas. (Not used at run-time).
G	Army2Global.xsl	Translates an Army message into a global message.
H	Navy2Global.xsl	Translates a Navy message into a global message.
I	Global2Army.xsl	Translates a global message into an Army message.
J	Global2Navy.xsl	Translates a global message into a Navy message.
K	Grid2LatLong.xsl	A stylesheet module.
L	LatLong2Grid.xsl	A stylesheet module.
M	NewGlobal.xml	An Army XML document that has been translated into a global XML document.
N	NewNavy.xml	An Army XML document that has been translated to a global, and then to a Navy XML document.
O	NewGlobal2.xml	A Navy XML document that has been translated into a global XML document.
P	NewArmy.xml	A Navy XML document that has been translated to a global, and then to a Navy XML document.
Q	Army2Global.xsl	Translates an Army Message into a Global message using Javascript commands

Table 4-3 Listing of files used in example

V. CONCLUSIONS

The purpose of our research was to find a means of communication between legacy systems, preferably using XML. While we were successful in the very limited demonstration of our consolidated type hierarchy, more work must be done to prove its applicability in C4ISR systems. This research was a first step, and should be followed by incorporating more functional transformations into the stylesheets, and then the application of the CTH to a set of real message formats.

The biggest advantage offered by the CTH is the reduction in the number of translations that must be defined. This advantage is realized by using a global, or bridge format for the various message types. Another significant benefit from the CTH model is the opportunity to automate part of the process of defining the translations. Automation could play a role at different stages in the generation of the stylesheets.

First, it is possible to create tool support for identifying elements in the new schema that correlate to an element in the global schema. [SC99] proposes a method for reconciling databases through semantic and structural matching. Since XML is a meta-language and is extensible, descriptive element names can be used, which lends itself to some level of syntactic matching between schemas. Since XML

also captures the structure of the data, structures can also be compared between schemas in order to find potential matches. Such a tool would identify possible matches in a graphical display, allow the engineer to confirm, override, or manually identify matches; and then make the appropriate entries in the global schema and CT.XML.

Another tool that would make the CTH easier to use is automated generation of the stylesheets. Once a message format has been mapped to the global schema, and the translations for individual elements have been identified in CT.XML, then the program should be able to automatically generate the stylesheets that translate entire messages to and from the global schema. All of the necessary information would be contained in the three documents of CT.XML, the global schema, and the system schema.

Another potential area for future work is to create a tool that would search a library of stylesheets in order to facilitate reuse of those transformations.

The best method of implementing the CTH may be in a publish/subscribe architecture. As the different systems log into the networked battlefield, the system would request to receive messages of a certain type. As each individual legacy system sends data over the network, a wrapper would intercept the message. The wrapper would mark up the message into a CTH XML representation, then send it to a web server. The web server would check the list of valid

subscribers for that message format, and send the message to those destinations. The destination system's XML wrapper would translate from the CTH mark-up form into the correct legacy system format.

By reutilizing the legacy systems similar to the mega-programming concept, we hope to save DoD thousands of dollars from cost savings and cost avoidance. Growing a Consolidated Type Hierarchy from our model will enable a variety of systems to communicate information across the battlefield regardless of branch or nationality.

The CTH is a powerful model that will allow more than just message systems to exchange information. It could be used for object-oriented databases, as well as source code files and initially any other kind of data. An application of this nature would allow more reuse of previously developed code and reduce development time and costs. An issue that remains to be investigated is the degree of overhead relative to real-time constraints and optimization methods for mitigating time and space overhead.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[DTIC] Defense Technical Information Center, Department of Defense Dictionary. http://www.dtic.mil/doctrine/jel/doddict/data/i/03090.html

[GW92] Wiederhold, G., Wegner, P., and Ceri, S. Toward Megaprogramming. Communications of the ACM (Nov. 1992).

[HMS] Hammer, J.; McLeod, D; Si, A.; Object Discovery and Unification in Federated Database Systems.

[JWJO00] Wing, J., and Ockerbloom, J. Respectful Type Converters. IEEE Transactions on Software Engineering (July 2000).

[KM98] Kahng, Jonghyun; McLeod, D. Dynamic Classificational Ontologies for Discovery in Cooperative Federated Databases. Cooperative Information Systems, 1996. Proceedings, First IFCIS International Conference on , 1996. Pages: 26 -35.

[MS] http://msdn.microsoft.com/downloads/default.asp?URL=/code/sample.asp?url=/MSDN-FILES/027/001/485/msdncompositedoc.xml

[MSDN] http://msdn.microsoft.com/library/default.asp?URL=/library/psdk/xmlsdk/xmlp7k6d.htm.

[MSDN2] http://msdn.microsoft.com/xml/XSLGuide/conformance.asp

[PROXML] Anderson, Richard; Birbeck, Mark; Kay, Michael; Livingstone, Steven; Loesgen, Brian; Martin, Didier; Mohr, Stephen; Ozu, Nicola; Peat, Bruce; Pinnock, Jonathon; Stark, Peter; Williams, Kevin. Professional XML. Wrox Press, August, 2000. p. 242.

[RKM] Roantree, M.; Keane, J.; Murphy, J. A Three-layer Model for Schema Management in Federated Databases. System Sciences, 1997, Proceedings of the Thirtieth Hwaii International Conference on , Volume: 1 , 1997 Pages: 44 -53 vol.1.

[SC99] S. Castano and V. De Antonellis. "A Schema Analysis and Reconciliation Tool Environment for Heteregeneous Databases", IEEE Databases, Feb 1999, pp. 53-62.

[Sin98] Narinder Singh. Unifying heterogeneous information models. Communications of the ACM 41, 5 (May. 1998), Pages 37 - 44.

[W3C] http://www.w3.org/TR/#About ©1995-2000, W3C.

BIBLIOGRAPHY

[AD99] van Deursen, T. Kuipers. "Identifying Objects using Cluster and Concept Analysis", Proceedings, 21st International Converference on Software Engineering, ICSE-99, ACM, 1999.

[CY94] C. Yu and W. Meng. "Progress in Database Search Strategies", IEEE Software, Oct 1994, pp. 1-19.

[EL99] E. Lee. "Embedded Software-An Agenda for Research", University of California Berkeley, Dec 1999. Available as UCB/ERL No. M99/63. (http://ptolemy.eecs.berkeley.edu/publications/papers/99/embedded/)

[DF91] D. Fang, J. Hammer, and D. McLeod. "The Identification and Resolution of Semantic Heterogeneity in Multidatabase Systems", IEEE Transactions on Software Engineering, Mar 1991, pp. 136-143.

[DH00] D. Hina. "Evaluation of the Extensible Mark-up Language as a Means for Establishing Interoperability between Heterogeneous Department of Defense (DoD) Databases." Naval Postgraduate School, Sep. 2000.

[HQDA] Headquarters, Dept. of the Army. ***Information Operations,* Field Manual 100-6,** Washington, DC: USGPO, 27 August 1996, p. iv.

[JBC] Joint Battle Center. "Joint Battle Management Integration(JBMI) Use Cases", Joint Battle Management Integration Assessment-Phase 2, 21 Aug 2000. *(internal document)* [MH99] Hiemstra, Michael A., Colonel, U.S. Army. Center for Army Lessons Learned, *NEWSLETTER NO. 99-2: "IO in a Peace Enforcement Environment"*, Fort Leavenworth, KS: USGPO, 19 June 1999.

[JH94a] J. Hammer, D. McLeod, and A. Si. "An Intelligent System for Identifying and Integrating Non-Local Objects in Federated Database Systems", University of Southern California. Available as a technical report on the internet. (ftp://ftp.usc.edu/pub/csinfo/tech-reports/papers/94-575.ps.Z)

[JH94b] J. Hammer, D. McLeod, and A. Si. "Object Discovery and Unification in Federated Database Systems", University of Southern California. Available as a technical report on the internet. (ftp://ftp.usc.edu/pub/csinfo/tech-reports/papers/94-574.ps.Z)

[L88] Luqi, V. Berzins, and R. Yeh. "A Prototyping Language for Real-Time Software", Software Engineering, IEEE Transactions on , Volume: 14 Issue: 10 , Oct. 1988, pp. 1409 -1423.

[TA00] T. Tran and J. Allen. "Interoperability and Security Support for Heterogeneous COTS/GOTS/Legacy Component Based Architecture. Naval Postgraduate School, Sep. 2000.

[VL99] V. Berzins, Luqi, B. Schultes, J. Guo, J. Allen, N Cheng, K. Gee, T. Nguyen, E. Stierna. "Interoperability Technology Assessment for Joint C4ISR Systems". Naval Postgraduate School, Sep. 2000.

APPENDIX A-ARMYMESSAGE.XML

This is the source file for the Army SALUTE message in XML. This was an input to translator along with a stylesheet, and was transformed into a global message, "NewGlobal.xml".

```
<!-- edited with XML Spy v3.5 NT beta 2 build Dec  1 2000 (http://www.xmlspy.com) by Brian Lyttle (Home) -->
<!--This file captures the representation of an Army SALUTE Report.  It is used when soldiers find an enemy on the battlefield, and
report the enemy's activity.  The Army constructed the report before automation, but today it still contains the same information.
The information is structured like this:
    S:  Size of the enemy unit, ie people, vehicles.
    A:  Activity of the enemy, ie walking, emplacing, sleeping.
    L:  Location in Military Grid Reference Position, with Grid identifier, Northing, and Easting.
    U:  Unit identification, to include distinctive symbols, patches, vehicle numbers.
    T:  Time the activity was observed.
    E:  Equipment the enemy possessed during the activity, such as M60 Machine Guns, AK-47s, mortars-->
<SOURCE name="ArmySystem" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=".\newSALUTESchema.xsd">
    <Type MsgID="SALUTE"/>
    <GroundUnit>
        <Size>10</Size>
        <Activity>WalkingNE</Activity>
        <Location>
            <GridID>NK</GridID>
            <Northing>100</Northing>
            <Easting>400</Easting>
        </Location>
        <UnitID>150MRR</UnitID>
        <Time>2159Z</Time>
        <Equipment>AK_47sampAT</Equipment>
        <DistanceInKms>10</DistanceInKms>
    </GroundUnit>
    <GroundUnit>
        <Size>5</Size>
        <Activity>RunningNE</Activity>
        <Location>
            <GridID>NK</GridID>
            <Northing>50</Northing>
            <Easting>350</Easting>
        </Location>
        <UnitID>100MRR</UnitID>
        <Time>2159Z</Time>
        <Equipment>M16</Equipment>
        <DistanceInKms>25</DistanceInKms>
    </GroundUnit>
</SOURCE>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B-NAVYMESSAGE.XML

This is the source file for the Navy Track Report message in XML. It shows what a Track Report would look like in XML.

```
<!--The Navy TrackReport possesses a set of tracks that identify objects. The objects are identified by a variety of
sensors such as Airborne radars and shipboard sensors. They communicate
information to each other via Tactical Data Links (TADIL) in a near real time fashion. The computers on-board the
sea and air platforms receive the infomration via the TADIL link, and use them in the information system as part of
a display for the operator. The display contains a picture of all nearby objects detected by the sensors. Our
representation is a simplified version used for our puposes to demonstrate the abilities of the CTH.
The entries for track are:
    Number: the number given to the object by the TADIL system.
    Coordinates: the latitude/longitude position of the object.
    Course: the direction (in degrees) of the object
    Speed: how fast the object is traveling in miles per hour
    Status: tells if the object is friendly, enemy, or unknown.
    IFF: the Identification Friend or Foe code that is received from the beacon on the object.
    GMT: time of the last sighting of this object, in Greenwich Mean Time.-->
<SOURCE name="NavyMessage" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=".\newTrackSchema.xsd">
    <Type MsgID="TrackReport"/>
    <Track>
        <Number>1000</Number>
        <Coordinates>
            <Latitude>32-36N</Latitude>
            <Longitude>30-20W</Longitude>
        </Coordinates>
        <Course>0</Course>
        <Speed>14</Speed>
        <Status>Unknown</Status>
        <IFF/>
        <GMT>1502</GMT>
        <DistanceInMiles>100</DistanceInMiles>
    </Track>
    <Track>
        <Number>1111</Number>
        <Coordinates>
            <Latitude>32-35N</Latitude>
            <Longitude>30-21W</Longitude>
        </Coordinates>
        <Course>0</Course>
        <Speed>14</Speed>
        <Status>Unknown</Status>
        <IFF/>
        <GMT>1503</GMT>
        <DistanceInMiles>10</DistanceInMiles>
    </Track>
</SOURCE>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C-SALUTESCHEMA.XSD

This is the XML Schema for the Army SALUTE Report, "SaluteSchema.xsd". It defines the structure of the "ArmyMessage.xml" document. This is the code represented by Figure 4-1.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT beta 2 build Dec  1 2000 (http://www.xmlspy.com) by Brian Lyttle (Home) -->
<!--W3C Schema generated by XML Spy v3.5 NT beta 2 build Dec  1 2000 (http://www.xmlspy.com)-->
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema" elementFormDefault="qualified">
    <xsd:element name="SOURCE">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Type">
                    <xsd:complexType>
                        <xsd:attribute name="MsgID" type="xsd:string" use="required"/>
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="GroundUnit" maxOccurs="unbounded">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="Size" type="xsd:byte"/>
                            <xsd:element name="Activity" type="xsd:string"/>
                            <xsd:element name="Location">
                                <xsd:complexType>
                                    <xsd:sequence>
                                        <xsd:element name="GridID" type="xsd:string"/>
                                        <xsd:element name="Northing" type="xsd:string"/>
                                        <xsd:element name="Easting" type="xsd:string"/>
                                    </xsd:sequence>
                                </xsd:complexType>
                            </xsd:element>
                            <xsd:element name="UnitID" type="xsd:string"/>
                            <xsd:element name="Time" type="xsd:string"/>
                            <xsd:element name="Equipment" type="xsd:string"/>
                            <xsd:element name="DistanceInKms" type="xsd:float"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D-TRACKSCHEMA.XSD

This is the XML Schema for the Navy Track Report, "TrackSchema.xsd". It defines the structure of "NavyMessage.xml". This is the code represented by Figure 4-1.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT beta 2 build Dec  1 2000 (http://www.xmlspy.com) by Brian Lyttle (Home) -->
<!--W3C Schema generated by XML Spy v3.5 NT beta 2 build Dec  1 2000 (http://www.xmlspy.com)-->
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema" elementFormDefault="qualified">
    <xsd:element name="SOURCE">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Type">
                    <xsd:complexType>
                        <xsd:attribute name="MsgID" type="xsd:string" use="required"/>
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="Track" maxOccurs="unbounded">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="Number" type="xsd:string"/>
                            <xsd:element name="Coordinates">
                                <xsd:complexType>
                                    <xsd:sequence>
                                        <xsd:element name="Latitude" type="xsd:string"/>
                                        <xsd:element name="Longitude" type="xsd:string"/>
                                    </xsd:sequence>
                                </xsd:complexType>
                            </xsd:element>
                            <xsd:element name="Course" type="xsd:string"/>
                            <xsd:element name="Speed" type="xsd:string"/>
                            <xsd:element name="Status" type="xsd:string"/>
                            <xsd:element name="IFF" type="xsd:string"/>
                            <xsd:element name="GMT" type="xsd:string"/>
                            <xsd:element name="DistanceInMiles" type="xsd:string"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E-GLOBALSCHEMA.XSD

This is the code from "GlobalSchema.xsd". It is represented by Figure 4-3. The global schema defines the structure of a global message, as in "NewGlobal.xml" and "NewGlobal2.xml".

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT beta 2 build Dec  1 2000 (http://www.xmlspy.com) by Brian Lyttle (Home) -->
<!--W3C Schema generated by XML Spy v3.5 NT beta 2 build Dec  1 2000 (http://www.xmlspy.com)-->
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema" elementFormDefault="qualified">
    <xsd:element name="SOURCE">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Type">
                    <xsd:complexType>
                        <xsd:attribute name="MsgID" type="xsd:string" use="required"/>
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="Track" maxOccurs="unbounded">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="Number" type="xsd:string"/>
                            <xsd:element name="GMT" type="xsd:string"/>
                            <xsd:element name="Location">
                                <xsd:complexType>
                                    <xsd:sequence>
                                        <xsd:element name="Latitude" type="xsd:string"/>
                                        <xsd:element name="Longitude" type="xsd:string"/>
                                    </xsd:sequence>
                                </xsd:complexType>
                            </xsd:element>
                            <xsd:element name="Status" type="xsd:string"/>
                            <xsd:element name="Course" type="xsd:string"/>
                            <xsd:element name="Speed" type="xsd:string"/>
                            <xsd:element name="IFF">
                                <xsd:complexType/>
                            </xsd:element>
                            <xsd:element name="Size" type="xsd:string"/>
                            <xsd:element name="Equipment" type="xsd:string"/>
                            <xsd:element name="DistanceInMiles" type="xsd:string"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX F-CT.XML

This file contains the relationships between the consolidated types found in the global schema and the elements found in the Army and Navy schemas. This is a concrete example of Figure 4-4.

```
<?xml version="1.0" encoding="UTF-8"?>
<ConsolidatedTypes xmlns="www.nps.navy.mil/sw/CTH/Global">
    <Track>
        <TrackReport name="Track" upXlate="Navy2Global.xsl" dnXlate="Global2Navy.xsl"/>
        <Salute name="GroundUnit" upXlate="Army2Global.xsl" dnXlate="Global2Army.xsl"/>
    </Track>
    <Number>
        <TrackReport name="Number"/>
        <Salute name="UnitID" upXlate="UnitID2Track.xsl" dnXlate="Track2UnitID.xsl"/>
    </Number>
    <Location>
        <TrackReport name="Location"/>
        <Salute name="Location" upXlate="Grid2LatLong.xsl" dnXlate="LatLong2Grid.xsl"/>
    </Location>
    <Course>
        <TrackReport name="Course"/>
    </Course>
    <Speed>
        <TrackReport name="Speed"/>
    </Speed>
    <Status>
        <TrackReport name="Status"/>
        <Salute name="Activity"/>
    </Status>
    <IFF>
        <TrackReport name="IFF"/>
    </IFF>
    <GMT>
        <TrackReport name="GMT"/>
        <Salute name="Time"/>
    </GMT>
    <Size>
        <Salute name="Size"/>
    </Size>
    <Equipment>
        <Salute name="Equipment"/>
    </Equipment>
    <Latitude>
        <TrackReport name="Latitude"/>
    </Latitude>
    <Longitude>
        <TrackReport name="Longitude"/>
    </Longitude>
    <DistanceInMiles>
        <TrackReport name="DistanceInMiles"/>
        <SALUTE name="DistanceInKms"/>
    </DistanceInMiles>
</ConsolidatedTypes>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G-ARMY2GLOBAL.XSL

This XSLT stylesheet transforms an Army SALUTE report into a global message. When we applied this stylesheet to "ArmyMessage.xml" the message produced was "NewGlobal.xml". Line numbers have been added to facilitate referral in the text.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet version="1.0" xmlns:xsl=http://www.w3.org/1999/XSL/Transform
3  xmlns:fo="http://www.w3.org/1999/XSL/Format">
4  <!--Stylesheet to translate from Army SALUTE Report to a CTH message-->

5  <xsl:import href=".\Grid2LatLong.xsl"/>

6  <xsl:template match = "/">
7      <xsl:apply-templates/>
8  </xsl:template>

9  <xsl:template match="SOURCE">
10     <SOURCE name="GlobalMessage" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
11                 xsi:noNamespaceSchemaLocation=".\GlobalSchema.xsd" >
12         <xsl:apply-templates/>
13     </SOURCE>
14 </xsl:template>

15 <xsl:template match="Type">
16     <Type MsgID="TrackReport"/>
17 </xsl:template>

18 <xsl:template match="GroundUnit">
19     <Track>
20         <xsl:apply-templates select="UnitID"/>
21         <xsl:apply-templates select="Time"/>
22         <xsl:apply-templates select="Location"/>
23         <xsl:apply-templates select="Activity"/>
24         <Course/>
25         <Speed/>
26         <IFF/>
27         <xsl:apply-templates select="Size"/>
28         <xsl:apply-templates select="Equipment"/>
29     </Track>
30 </xsl:template>

31 <xsl:template match="UnitID">
32     <Number>
33         <xsl:value-of select="."/>
34     </Number>
35 </xsl:template>

36 <xsl:template match="Time">
37     <GMT>
38         <xsl:value-of select="."/>
39     </GMT>
40 </xsl:template>
```

```
41 <xsl:template match="Location">
42     <Location>
43     <xsl:apply-templates/>
44     </Location>
45 </xsl:template>

46 <xsl:template match="Activity">
47     <Status>
48         <xsl:value-of select="."/>
49     </Status>
50 </xsl:template>

51 <xsl:template match="Size">
52     <Size>
53         <xsl:value-of select="."/>
54     </Size>
55 </xsl:template>

56 <xsl:template match="Equipment">
57     <Equipment>
58         <xsl:value-of select="."/>
59     </Equipment>
60 </xsl:template>

61 </xsl:stylesheet>
```

APPENDIX H-NAVY2GLOBAL.XSL

This XSLT stylesheet transforms a Navy Track report into a global message. When we applied this stylesheet to "NavyMessage.xml" the message produced was "NewGlobal2.xml".

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <!--Stylesheet to translate from a Navy Track Report to a CTH message-->

    <xsl:template match = "/">
        <xsl:apply-templates/>
    </xsl:template>

    <xsl:template match="SOURCE">
        <SOURCE name="GlobalMessage" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation=".\newGlobalSchema.xsd" >
            <xsl:apply-templates/>
        </SOURCE>
    </xsl:template>

    <xsl:template match="Type">
        <Type MsgID="TrackReport"/>
    </xsl:template>

    <xsl:template match="Track">
        <Track>
            <xsl:apply-templates select="Number"/>
            <xsl:apply-templates select="GMT"/>
            <xsl:apply-templates select="Coordinates"/>
            <xsl:apply-templates select="Status"/>
            <xsl:apply-templates select="Course"/>
            <xsl:apply-templates select="Speed"/>
            <xsl:apply-templates select="IFF"/>
            <Size/>
            <Equipment/>
            <xsl:apply-templates select="DistanceInMiles"/>
        </Track>
    </xsl:template>

    <xsl:template match="Number">
        <Number>
            <xsl:value-of select="."/>
        </Number>
    </xsl:template>

    <xsl:template match="Coordinates">
        <Location>
            <xsl:apply-templates/>
        </Location>
    </xsl:template>

    <xsl:template match="Latitude">
        <Latitude>
            <xsl:apply-templates/>
```

```
        </Latitude>
    </xsl:template>

    <xsl:template match="Longitude">
        <Longitude>
            <xsl:apply-templates/>
        </Longitude>
    </xsl:template>

    <xsl:template match="Course">
        <Course>
            <xsl:value-of select="."/>
        </Course>
    </xsl:template>

    <xsl:template match="Speed">
        <Speed>
            <xsl:value-of select="."/>
        </Speed>
    </xsl:template>

    <xsl:template match="Status">
        <Status>
            <xsl:value-of select="."/>
        </Status>
    </xsl:template>

    <xsl:template match="IFF">
        <IFF>
            <xsl:value-of select="."/>
        </IFF>
    </xsl:template>

    <xsl:template match="GMT">
        <GMT>
            <xsl:value-of select="."/>
        </GMT>
    </xsl:template>

    <xsl:template match="DistanceInMiles">
        <DistanceInMiles>
            <xsl:value-of select="."/>
        </DistanceInMiles>
    </xsl:template>


</xsl:stylesheet>
```

APPENDIX I-GLOBAL2ARMY.XSL

This XSLT stylesheet transforms a global message into an Army SALUTE report. When we applied this stylesheet to "NewGlobal.xml" the message produced was "NewArmy.xml".

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/TR/WD-xsl" language="JavaScript">
    <!--Stylesheet to translate from a CTH message to an Army SALUTE Report-->

<!-- <xsl:import href=".\LatLong2Grid.xsl"/>-->

    <xsl:template match = "/">
        <xsl:apply-templates/>
    </xsl:template>

    <xsl:template match="SOURCE">
        <SOURCE name="ArmySystem" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
                    xsi:noNamespaceSchemaLocation=".\newSALUTEschema.xsd" >
            <xsl:apply-templates/>
        </SOURCE>
    </xsl:template>

    <xsl:template match="Type">
        <Type MsgID="SALUTE"/>
    </xsl:template>

    <xsl:template match="Track">
        <GroundUnit>
            <xsl:apply-templates select="Size"/>
            <xsl:apply-templates select="Status"/>
            <xsl:apply-templates select="Location"/>
            <xsl:apply-templates select="Number"/>
            <xsl:apply-templates select="GMT"/>
            <xsl:apply-templates select="Equipment"/>
            <xsl:apply-templates select="DistanceInMiles"/>
        </GroundUnit>
    </xsl:template>

    <xsl:template match="Number">
        <UnitID>
            <xsl:value-of select="."/>
        </UnitID>
    </xsl:template>

    <xsl:template match="GMT">
        <Time>
            <xsl:value-of select="."/>
        </Time>
    </xsl:template>

    <xsl:template match="Location">
        <Location>
            <xsl:apply-templates/>
        </Location>
    </xsl:template>
```

```
    <xsl:template match="Latitude">
        <GridID>
        </GridID>
        <Northing>
            <xsl:value-of select="."/>
        </Northing>
    </xsl:template>

    <xsl:template match="Longitude">
        <Easting>
            <xsl:value-of select="."/>
        </Easting>
    </xsl:template>

    <xsl:template match="Status">
        <Activity>
            <xsl:value-of select="."/>
        </Activity>
    </xsl:template>

    <xsl:template match="Size">
        <Size>
            <xsl:value-of select="."/>
        </Size>
    </xsl:template>

    <xsl:template match="Equipment">
        <Equipment>
            <xsl:value-of select="."/>
        </Equipment>
    </xsl:template>

    <xsl:template match="DistanceInMiles">
        <DistanceInKms><xsl:eval>this.nodeTypedValue*(2.21)</xsl:eval></DistanceInKms>
    </xsl:template>

</xsl:stylesheet>
```

APPENDIX J-GLOBAL2NAVY.XSL

This XSLT stylesheet transforms a global message into a Navy Track report. When we applied this stylesheet to "NewGlobal2.xml" the message produced was "NewNavy.xml".

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <!--Stylesheet to translate from a CTH message to a Navy Track Report-->

    <xsl:template match = "/">
        <xsl:apply-templates/>
    </xsl:template>

    <xsl:template match="SOURCE">
        <SOURCE name="NavyMessage"
            xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation=".\newTrackSchema.xsd" >
            <xsl:apply-templates/>
        </SOURCE>
    </xsl:template>

    <xsl:template match="Type">
        <Type MsgID="TrackReport"/>
    </xsl:template>

    <xsl:template match="Track">
        <Track>
            <xsl:apply-templates select="Number"/>
            <xsl:apply-templates select="Location"/>
            <xsl:apply-templates select="Course"/>
            <xsl:apply-templates select="Speed"/>
            <xsl:apply-templates select="Status"/>
            <xsl:apply-templates select="IFF"/>
            <xsl:apply-templates select="GMT"/>
            <xsl:apply-templates select="DistanceInMiles"/>
        </Track>
    </xsl:template>

    <xsl:template match="Number">
        <Number>
            <xsl:value-of select="."/>
        </Number>
    </xsl:template>

    <xsl:template match="Location">
        <Coordinates>
            <xsl:apply-templates/>
        </Coordinates>
    </xsl:template>

    <xsl:template match="Latitude">
        <Latitude>
            <xsl:apply-templates/>
        </Latitude>
```

```
    </xsl:template>

    <xsl:template match="Longitude">
        <Longitude>
            <xsl:apply-templates/>
        </Longitude>
    </xsl:template>

    <xsl:template match="Course">
        <Course>
            <xsl:value-of select="."/>
        </Course>
    </xsl:template>

    <xsl:template match="Speed">
        <Speed>
            <xsl:value-of select="."/>
        </Speed>
    </xsl:template>

    <xsl:template match="Status">
        <Status>
            <xsl:value-of select="."/>
        </Status>
    </xsl:template>

    <xsl:template match="IFF">
        <IFF>
            <xsl:value-of select="."/>
        </IFF>
    </xsl:template>

    <xsl:template match="GMT">
        <GMT>
            <xsl:value-of select="."/>
        </GMT>
    </xsl:template>

    <xsl:template match="DistanceInMiles">
        <DistanceInMiles>
            <xsl:value-of select="."/>
        </DistanceInMiles>
    </xsl:template>


</xsl:stylesheet>
```

APPENDIX K-GRID2LATLONG.XSL

This XSLT stylesheet is imported by "Army2Global.xsl". This stylesheet does not actually convert a grid position into a latitude-longitude position. We used this stylesheet to test and demonstrate the modularity of XSLT stylesheets.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">


        <xsl:template match="GridID">
        </xsl:template>

        <xsl:template match="Northing">
            <Latitude>
                <xsl:value-of select="."/>
            </Latitude>
        </xsl:template>

        <xsl:template match="Easting">
            <Longitude>
                <xsl:value-of select="."/>
            </Longitude>
        </xsl:template>

</xsl:stylesheet>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX L-LATLONG2GRID.XSL

This XSLT stylesheet is imported by "Global2Army.xsl". This stylesheet does not actually convert a latitude-longitude position into a grid position. We used this stylesheet to test and demonstrate the modularity of XSLT stylesheets.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">

    <xsl:template match="Latitude">
        <GridID>
        </GridID>
        <Northing>
            <xsl:value-of select="."/>
        </Northing>
    </xsl:template>

    <xsl:template match="Longitude">
        <Easting>
            <xsl:value-of select="."/>
        </Easting>
    </xsl:template>

</xsl:stylesheet>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX M-NEWGLOBAL.XML

This is the output of the XSL processor when "Army2Global.xsl" is applied to "ArmyMessage.xml"

```
<SOURCE name="GlobalMessage" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=".\newGlobalSchema.xsd">
    <Type MsgID="TrackReport"/>
    <Track>
        <Number>
            150MRR
        </Number>
        <GMT>
            2159Z
        </GMT>
        <Location>
            <Latitude>
                100
            </Latitude>
            <Longitude>
                400
            </Longitude>
        </Location>
        <Status>
            WalkingNE
        </Status>
        <Course/>
        <Speed/>
        <IFF/>
        <Size>
            10
        </Size>
        <Equipment>
            AK_47sampAT
        </Equipment>
        <DistanceInMiles>
            4.52488687782805
        </DistanceInMiles>
    </Track>
    <Track>
        <Number>
            100MRR
        </Number>
        <GMT>
            2159Z
        </GMT>
        <Location>
            <Latitude>
                50
            </Latitude>
            <Longitude>
                350
            </Longitude>
        </Location>
        <Status>
            RunningNE
        </Status>
        <Course/>
```

```
        <Speed/>
        <IFF/>
        <Size>
            5
        </Size>
        <Equipment>
            M16
        </Equipment>
        <DistanceInMiles>
            11.3122171945701
        </DistanceInMiles>
    </Track>
</SOURCE>
```

APPENDIX N-NEWNAVY.XML

This is the output of the XSL processor when "Global2Navy.xsl" is applied to "NewGlobal.xml"

```
<SOURCE name="NavyMessage" xsi:noNamespaceSchemaLocation=".\newTrackSchema.xsd"
xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance">
    <Type MsgID="TrackReport"/>
    <Track>
        <Number>
            150MRR
        </Number>
        <Coordinates>
            <Latitude>
            100
        </Latitude>
            <Longitude>
            400
        </Longitude>
        </Coordinates>
        <Course/>
        <Speed/>
        <Status>
            WalkingNE
        </Status>
        <IFF/>
        <GMT>
            2159Z
        </GMT>
        <DistanceInMiles>
            4.52488687782805
        </DistanceInMiles>
    </Track>
    <Track>
        <Number>
            100MRR
        </Number>
        <Coordinates>
            <Latitude>
            50
        </Latitude>
            <Longitude>
            350
        </Longitude>
        </Coordinates>
        <Course/>
        <Speed/>
        <Status>
            RunningNE
        </Status>
        <IFF/>
        <GMT>
            2159Z
        </GMT>
        <DistanceInMiles>
            11.3122171945701
        </DistanceInMiles>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX O-NEWGLOBAL2.XML

This is the output of the XSL processor when "Navy2Global.xsl" is applied to "NavyMessage.xml"

```
<?xml version="1.0" encoding="UTF-16"?>
<SOURCE name="GlobalMessage" xsi:noNamespaceSchemaLocation=".\newGlobalSchema.xsd"
xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance">
    <Type MsgID="TrackReport"/>
    <Track>
        <Number>1000</Number>
        <GMT>1502</GMT>
        <Location>
            <Latitude>32-36N</Latitude>
            <Longitude>30-20W</Longitude>
        </Location>
        <Status>Unknown</Status>
        <Course>0</Course>
        <Speed>14</Speed>
        <IFF/>
        <Size/>
        <Equipment/>
        <DistanceInMiles>100</DistanceInMiles>
    </Track>
    <Track>
        <Number>1111</Number>
        <GMT>1503</GMT>
        <Location>
            <Latitude>32-35N</Latitude>
            <Longitude>30-21W</Longitude>
        </Location>
        <Status>Unknown</Status>
        <Course>0</Course>
        <Speed>14</Speed>
        <IFF/>
        <Size/>
        <Equipment/>
        <DistanceInMiles>10</DistanceInMiles>
    </Track>
</SOURCE>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX P-NEWARMY.XML

This is the output of the XSL processor when "Global2Navy.xsl" is applied to "NewGlobal.xml"

```
<SOURCE name="ArmySystem" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=".\newSALUTEschema.xsd">
    <Type MsgID="SALUTE"/>
    <GroundUnit>
        <Size/>
        <Activity>
            Unknown
        </Activity>
        <Location>
            <GridID/>
            <Northing>
            32-36N
        </Northing>
            <Easting>
            30-20W
        </Easting>
        </Location>
        <UnitID>
            1000
        </UnitID>
        <Time>
            1502
        </Time>
        <Equipment/>
        <DistanceInKms>221</DistanceInKms>
    </GroundUnit>
    <GroundUnit>
        <Size/>
        <Activity>
            Unknown
        </Activity>
        <Location>
            <GridID/>
            <Northing>
            32-35N
        </Northing>
            <Easting>
            30-21W
        </Easting>
        </Location>
        <UnitID>
            1111
        </UnitID>
        <Time>
            1503
        </Time>
        <Equipment/>
        <DistanceInKms>22.1</DistanceInKms>
    </GroundUnit>
</SOURCE>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX Q-ARMY2GLOBAL.XSL USING ''XSL:EVAL''

This is file differs from Appendix G because it uses the "xsl:eval" command and does not use the import ability implemented in the w3c version of XSL. However, it does convert from kilometers to miles and still transforms MGRS to lat/long coordinates.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/TR/WD-xsl" language="JavaScript">
    <!--Stylesheet to translate from Army SALUTE Report to a CTH message-->
    <!-- <xsl:include href=".\Grid2LatLong.xsl"/>-->
    <!--The include statement is an accepted statement in a different XSL namespace called
xmlns:xsl="http://www.w3.org/1999/XSL/Transform".
However, in the namespace used by this stylesheet, "include" and "import" are not accepted commands.  Since we
wanted to demonstrate the
ability of XML to functionally transform objects, we selected the above namespace.  The "XSL/Transform"
namespace is used to transform the
trees formed by the two documents, while the "TR/WD-xsl" namespace is used to format objects for a destination
system.

The w3c is reviewing different recommendations, and we hope the two namespaces are combined :).-->
    <xsl:template match="/">
        <xsl:apply-templates/>
    </xsl:template>
    <xsl:template match="SOURCE">
        <SOURCE name="GlobalMessage"
            xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
                    xsi:noNamespaceSchemaLocation=".\newGlobalSchema.xsd" >
            <xsl:apply-templates/>
        </SOURCE>
    </xsl:template>
    <xsl:template match="Type">
        <Type MsgID="TrackReport"/>
    </xsl:template>
    <xsl:template match="GroundUnit">
        <Track>
            <xsl:apply-templates select="UnitID"/>
            <xsl:apply-templates select="Time"/>
            <xsl:apply-templates select="Location"/>
            <xsl:apply-templates select="Activity"/>
            <Course/>
            <Speed/>
            <IFF/>
            <xsl:apply-templates select="Size"/>
            <xsl:apply-templates select="Equipment"/>
            <xsl:apply-templates select="DistanceInKms"/>
        </Track>
    </xsl:template>
    <xsl:template match="UnitID">
        <Number>
            <xsl:value-of select="."/>
        </Number>
    </xsl:template>
    <xsl:template match="Time">
        <GMT>
            <xsl:value-of select="."/>
        </GMT>
    </xsl:template>
```

```
    <xsl:template match="Location">
        <Location>
            <xsl:apply-templates/>
        </Location>
    </xsl:template>
    <xsl:template match="Activity">
        <Status>
            <xsl:value-of select="."/>
        </Status>
    </xsl:template>
    <xsl:template match="Size">
        <Size>
            <xsl:value-of select="."/>
        </Size>
    </xsl:template>
    <xsl:template match="Equipment">
        <Equipment>
            <xsl:value-of select="."/>
        </Equipment>
    </xsl:template>
    <xsl:template match="GridID"/>
    <xsl:template match="Northing">
        <Latitude>
            <xsl:value-of select="."/>
        </Latitude>
    </xsl:template>
    <xsl:template match="Easting">
        <Longitude>
            <xsl:value-of select="."/>
        </Longitude>
    </xsl:template>
    <xsl:template match="DistanceInKms">
        <DistanceInMiles>
            <xsl:eval>this.nodeTypedValue/(2.21)</xsl:eval>
        </DistanceInMiles>
    </xsl:template>
</xsl:stylesheet>
```

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, California 93943-5101

3. Engineering & Technology Curriculum 1
   Code 34
   Naval Postgraduate School
   700 Dyer Rd., Room 115
   Monterey, California 93943-5107

4. Computer and Information Programs Office 1
   Code 32
   Naval Postgraduate School
   833 Dyer Rd., Room 404
   Monterey, California 93943-5120

5. Dr Luqi 1
   Naval Postgraduate School
   833 Dyer Rd.
   Monterey, California 93943-5118

5. Dr Valdis Berzins 2
   Naval Postgraduate School
   833 Dyer Rd.
   Monterey, California 93943-5118

6. Dr Ge Jun 2
   Naval Postgraduate School
   833 Dyer Rd.
   Monterey, California 93943-5118

7. Captain Paul E. Young 1
   Naval Postgraduate School
   833 Dyer Rd.
   Monterey, California 93943-5118

8. Lieutenant Todd P. Ehrhardt 2
   388 Woodhams Rd.
   Santa Clara, CA 95051

9. Captain Brian J. Lyttle.....2
   7321 East 66th Pl.
   Tulsa, Oklahoma 74133

10. Professor Dan Boger.....1
    Naval Postgraduate School
    833 Dyer Rd.
    Monterey, California 93943-5118